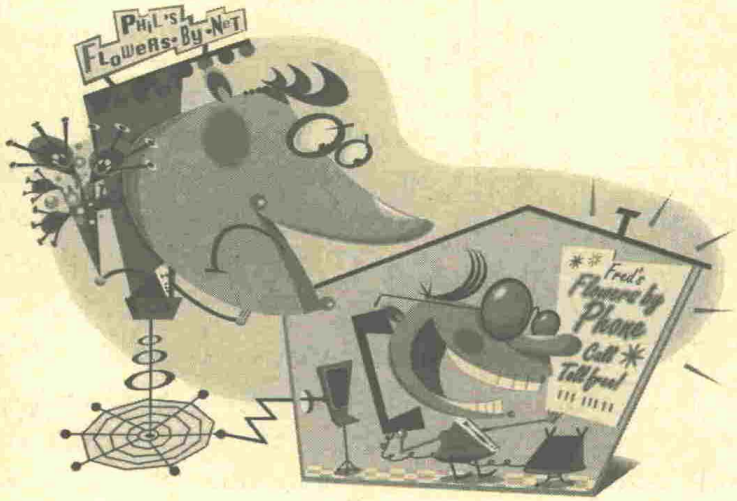
AutoCAD

ชัชวาล ศุภเกษม

เทคนิคการใช้ AutoCAD : เขียนชุดอักษร หรือตัวเลขเรียง



สิ่งที่ต้องการพื้นฐาน

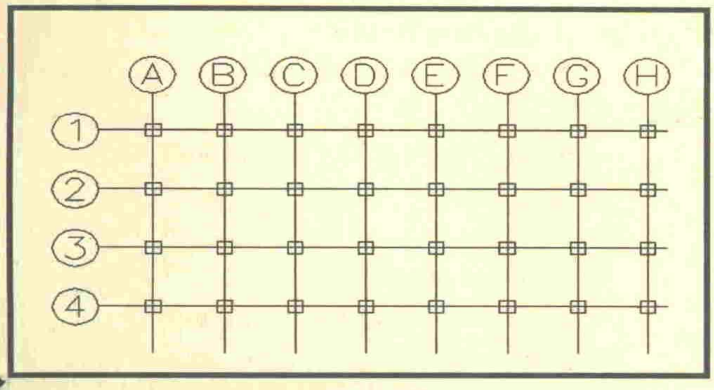
- โปรแกรม AutoCAD รีลีส 12 ขึ้นไป ทั้งฉบับดอส หรือวินโดวส์
- โปรแกรม Text editor ที่สามารถบันทึกเป็นไฟล์รหัสแอสกีได้ เช่น Edit.com ใน MS-DOS เวอร์ชัน 5 ขึ้นไป, Write หรือ Thai Write ในไมโครซอฟต์วินโดวส์ โดยเลือก save ในแบบ Word for DOS/ Txt Only, Notepad ในไมโครซอฟต์วินโดวส์
- มีพื้นฐานการเขียนแบบด้วย AutoCAD พอสมควร

สร้างคำสั่งไว้ช่วยงานเขียน เพื่อเขียนชุดอักษร หรือตัวเลขเรียง

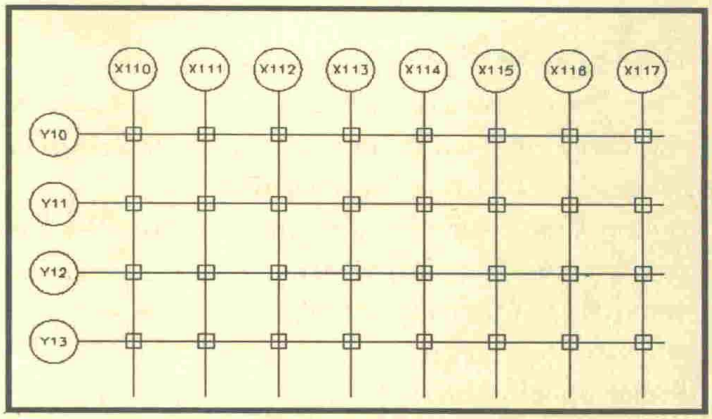
การใช้งาน AutoCAD เพื่อเขียนตัวอักษร สามารถเลือกกำหนดตัวอักษรที่กำลังจะเขียนให้มีรูปร่างต่าง ๆ โดยอาศัยคำสั่ง STYLE ในการกำหนดโดยเลือกใช้ไฟล์ฟอนต์ ซึ่งเป็นตัวกำหนดทิศทางการวาดเส้นเป็นรูปตัวอักษรหนึ่ง ๆ และในคำสั่งสามารถกำหนดคุณสมบัติเพิ่มเติมแก่รูปตัวอักษร เช่น กำหนดสัดส่วนความกว้าง มุมเอียง กลับซ้าย-ขวา บน-ล่าง หรือกำหนดให้ถูกเขียนเรียงแบบแนวดิ่ง แทนที่จะถูกเขียนเรียงในแนวนอนแบบปกติทั่วไป หรือแม้แต่กำหนดให้เป็นแบบความสูงคงที่ค่าหนึ่ง ไม่ให้มีพรอมต์กำหนดความสูง ในคำสั่งที่ใช้เขียนอักษร ซึ่งได้แก่ TEXT และ DTEXT และสำหรับรีลีส 13 ยังได้เพิ่มความสามารถของการเขียนข้อความแบบทีละย่อหน้า หรือ Paragraph ด้วยคำสั่ง MTEXT (Multiline text)

ในงานเขียนแบบไม่ว่าจะเป็นสาขาใด ๆ มักจะมีส่วนที่เกี่ยวข้องกับการเขียนตัวอักษรแบบเรียง (A, B, C, ...) ชุดเลขเรียง (1, 2, 3, ...) หรือเลขเรียงที่มีอักษรนำหน้า (T01, T02, T03, ...) เช่น แบบแปลนงานก่อสร้างจะมีการกำหนดหมายเลขของเสาเพื่อกำหนดตำแหน่งไว้อ้างอิง ที่เรียกว่า Grid line ทั้งที่มี Prefix และไม่มีก่อนหมายเลข เขียนเรียงอย่างสม่ำเสมอ (Uniform) ดูรูปที่ 1 และ 2 แบบตารางโหลดของงานไฟฟ้า เขียนหมายเลขเรียงสลับซ้าย-ขวา ดังในรูปที่ 3 แบบประกอบ (Assembly Drawing) ของชิ้นงานเครื่องกลที่ต้องมีหมายเลขกำกับแต่ละชิ้นส่วน ซึ่งเขียนกำกับกระจายทั่วไป (Random) ในแบบ พร้อมกับชุดตัวเลขเรียงอย่างสม่ำเสมอในตารางกำกับรายละเอียด ดูรูปที่ 4 และเลขเรียงสำหรับกำหนดหมายเลขขั้วต่อสาย (Terminal) ตามรูปที่ 5

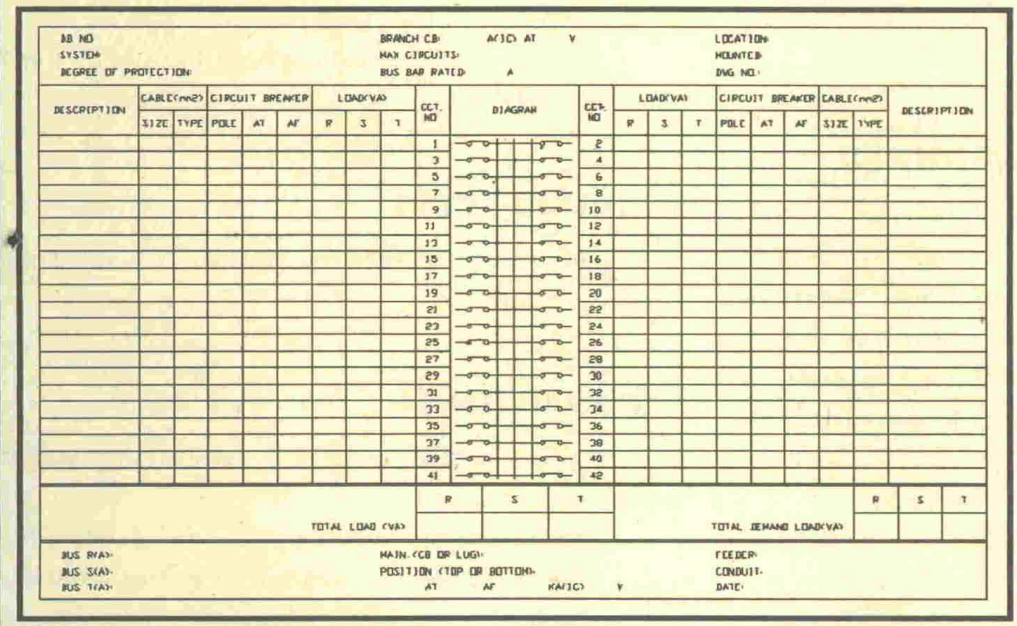
ในการเขียนข้อความที่มีลักษณะเรียงค่าเพิ่มทีละหนึ่งขั้น (Incrementing Text) เช่นนี้ ใน AutoCAD ไม่ได้มีคำสั่งพิเศษใดที่ช่วยกำหนดค่าเองอย่างอัตโนมัติ เราต้องทำการป้อนค่าทีละหนึ่งรายการไปจนครบ หรือเราอาจใช้การทำซ้ำ (Copy) ไปยังตำแหน่งที่ต้องการ แล้วค่อยรันคำสั่ง DDEDIT เพื่อแก้ไขให้ค่าตรงตามต้องการ เพื่อให้การจัดการงานลักษณะนี้เป็นไปได้โดยสะดวก บทความนี้จะนำเสนอคำสั่ง TXINC ที่ผู้เขียนพัฒนาขึ้นจากการโปรแกรม AutoLISP ให้เป็นทางเลือกใหม่



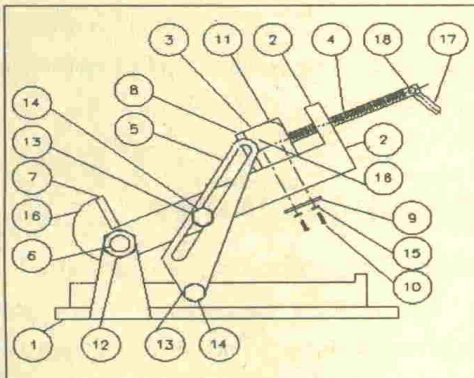
รูปที่ 1 แสดงการประยุกต์ใช้ในงานเขียนชุดตัวเลขเรียง แบบแปลนงานก่อสร้าง



รูปที่ 2 แสดงการประยุกต์ใช้ในงานเขียนชุดตัวเลขเรียง แบบแปลนงานก่อสร้าง ขอให้สังเกตว่ามี Prefix (X และ Y) ในชุดเลขเรียงนี้ด้วย

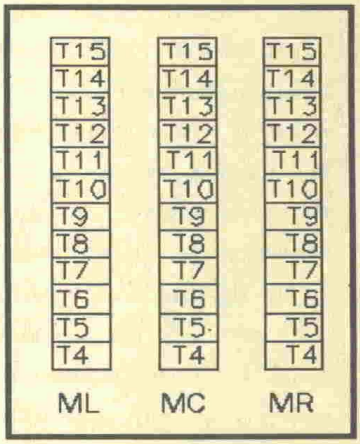


รูปที่ 3 แสดงการประยุกต์ใช้ในงานเขียนชุดตัวเลขเรียง



18	RD HD RIVET	1	STEEL
17	HANDLE	1	STEEL
16	RD HD CAP	4	STEEL
15	LOCK WASHER	2	STEEL
14	PL WASHER	4	STEEL
13	HEX HD CAP	4	STEEL
12	HEX HD NUT	2	STEEL
11	SET SCREW	1	STEEL
10	RD HD SCR	2	STEEL
9	JAW HOLDER	1	STEEL
8	JAW PLATE	1	STEEL
7	BASE PLATE	1	STEEL
6	SHAFT	1	STEEL
5	HINGE	2	STEEL
4	SCREW	1	STEEL
3	SLIDING JAW	1	CAST
2	MOVABLE BASE	1	CAST
1	BASE	1	CAST

รูปที่ 4 แสดงการประยุกต์ใช้ในงานเขียนชุดตัวเลขเรียงแบบ Random



รูปที่ 5 แสดงการประยุกต์ใช้ในงานเขียนชุดตัวเลขเรียง เมื่อกำหนด Justify เป็น ML, MC และ MR

คำสั่ง TXINC

คำสั่ง TXINC (TeXt INCrement) มีความสามารถในการช่วยเขียนชุดอักษรหรือตัวเลขเรียงได้ การทำงานของคำสั่งนี้จะเริ่มจากให้ผู้ใช้ป้อนค่าเบื้องต้นที่จำเป็นที่จะใช้ในการเขียน ได้แก่

- Justify ตำแหน่งของการเขียน เหมือนกับในคำสั่ง TEXT ปกติ กำหนดแบบเขียนชิดซ้าย ชิดขวา หรือกึ่งกลาง ฯ
- ค่าเริ่มตัวแรกที่จะเขียน (ย้ำว่าเฉพาะค่าตัวแรก, First value)
- อักษรนำหน้า (Prefix)
- อักษรตามหลัง (Suffix)
- ความสูงตัวอักษร (Height)
- จากนั้นจะให้กำหนดวิธีการกำหนดตำแหน่งของการวางข้อความเหล่านี้ ซึ่งทำได้ 2 แบบคือ

1. แบบสม่ำเสมอ (Uniform)
2. แบบกระจายทั่วไป (Random)

แบบสม่ำเสมอ และกระจายทั่วไป

ถ้ากำหนดให้คำสั่งนี้ ทำงานกำหนดตำแหน่งแบบกระจายทั่วไป จะไม่ต้องป้อนค่าเบื้องต้นอะไรอีก คำสั่งจะมีพรอมต์ให้กำหนดจุดเพื่อเขียนได้เลย (Draw point:) ส่วนค่าตัวอักษร หรือตัวเลข คำสั่งจะจัดการกำหนดให้เอง เรียงลำดับเพิ่มทีละหนึ่งขั้นจากค่าตัวแรก (ค่าที่ตอบในพรอมต์ First value) เมื่อเขียนเสร็จในตำแหน่งแรกแล้วจะยังคงวนมาที่พรอมต์ให้กำหนดจุดเขียนต่อไป เรียงเพิ่มจากค่าก่อนหน้า เป็นเช่นนี้ไปจนกว่าจะมีการกด Enter แทนการป้อนค่าจุด กรณีเช่นนี้ ได้แก่ การกำหนดหมายเลขอ้างอิงในแบบทั่ว ๆ ไป เช่นในรูปที่ 4

ส่วนในกรณีกำหนดแบบสม่ำเสมอ กล่าวคือ มีระยะห่างที่เท่ากันในแต่ละข้อความ รวมถึงมีทิศทางเดียวกันด้วย จะให้ผู้ใช้ป้อนค่าที่จะใช้อิงในการเขียนอีกดังนี้

- ทิศทาง หรือมุมของแต่ละข้อความ (Alignment:)
- จำนวนข้อความที่จะเขียน (No of Text:)
- ระยะห่างของแต่ละข้อความ (Distance:)
- จุดเริ่มต้นของการเขียน (Start point:)

สำหรับจุดเริ่มต้นของการเขียนจะป้อนค่าเพียงครั้งเดียว ที่เหลือคำสั่งจะคำนวณให้เอง โดยอาศัยค่าที่ได้กำหนดไว้ในข้างต้น สำหรับในกรณีนี้ได้แก่ การเขียนหมายเลขกำกับของขั้วต่อสาย (Terminal) หมายเลขของ Grid line ในแบบที่ห่างกันสม่ำเสมอ

จุดเด่นของคำสั่ง

1. เขียนชุดอักษรเรียง เช่น

A, B, D, ... หรืออาจเริ่มต้นที่อักษรใด ๆ ก็ได้ เช่น M, N, O, ...

2. เขียนชุดตัวเลขเรียง เช่น

1, 2, 3, ... หรืออาจเริ่มต้นที่ตัวเลขใด ๆ ก็ได้ เช่น 100, 101 102, ...

3. มี Prefix หรืออักษรนำหน้าได้ กี่ตัวอักษรก็ได้ เช่น

A1, A2, A3, ...

T01, T02, T03, ...

WORK01, WORK02, WORK03, ...

4. มี Suffix หรืออักษรลงท้ายได้ กี่ตัวอักษรก็ได้ เช่น

T1E, T2E, T3E, ...

T1MN, T2MN, T3MN, ...

5. คำสั่งตรวจสอบค่าเริ่มต้น (First value) ที่ป้อนเข้ามาว่าเป็นตัวอักษร หรือตัวเลขเอง (Alpha or Numeric)

6. เลือกทำงานได้สองแบบในการกำหนดตำแหน่ง คือแบบสม่ำเสมอ และกระจายทั่วไป

7. จำค่าที่กำหนดไว้ในครั้งก่อนมาเป็นค่าดีฟอลต์ต่อมา (เมื่อยังไม่มีการโหลดไฟล์ใหม่)

พรอมต์ของคำสั่ง

เมื่อรันคำสั่ง TXINC ที่บรรทัด Command: ใน AutoCAD จะมีพรอมต์ ดังตารางที่ 1

สิ่งที่ต้องคำนึงถึง

1. Text Style ที่ใช้ในการเขียนเป็นตัวเดียวกับที่กำหนดไว้ในปัจจุบัน (Current Text Style)

2. ในคำสั่งมีพรอมต์ถามความสูง (Height:) ดังนั้นต้องแน่ใจว่า Text Style ปัจจุบันไม่ได้ถูกกำหนดไว้แบบความสูงคงค่า (Fixed Text Height) หรือตอนสร้าง Style ให้กำหนดที่พรอมต์ถามความสูงเป็นศูนย์

3. กำหนด Layer ปัจจุบันให้ตรงตามที่ต้องการก่อนลงมือเขียน ซึ่งเป็นสิ่งปกติที่พึงปฏิบัติในการเขียนแบบด้วย AutoCAD อยู่แล้ว

4. ที่พรอมต์ถามความสูง (Height:) ค่าที่ป้อนต้องคำนึงถึงสัดส่วนหรือสเกลที่ใช้ด้วย

การกำหนดความสูงตัวอักษร

AutoCAD ตั้งแต่รีลีส 11 เป็นต้นมามีการนำเสนอแนวคิดใหม่ในการทำงานเป็น 2 ภาวะ คือ ภาวะ Model space และ Paper space ซึ่งช่วยให้การจัดการในการพิมพ์แบบที่เกี่ยวข้องกับการกำหนดสเกลออกทางเครื่องพิมพ์เป็นไปอย่างง่ายดาย ต่อไปนี้จะเป็นข้อสรุปสั้น ๆ เกี่ยวกับการกำหนดขนาดในการทำงานทั้งสองภาวะนี้ โดยจะถือว่าแบบที่พิจารณาถูกสร้างในหน่วยเมตรและมิลลิเมตรเท่านั้น

ตารางที่ 1

Command: พิมพ์ TXINC ↵	
Center/Middle/Right/TL/TC/TR/ML/MC/MR/BL/BC/BR:	♦ กำหนด Justify แบบในคำสั่ง TEXT หรือ DTEXT
First value: <>	♦ ค่าเริ่มต้น
Prefix: <>	♦ ตัวอักษรนำหน้า (เป็นคำได้)
Suffix: <>	♦ ตัวอักษรลงท้าย (เป็นคำได้)
Height: <>	♦ ความสูงตัวอักษร
Uniform/Random points:	♦ เลือกกำหนดวิธีการวางตำแหน่ง
ถ้าเลือก R (Random points) จะมีพรอมต์ ดังนี้	
Draw point:	♦ เลือกจุดที่ต้องการเขียนไปเรื่อย จนครบแล้วกด ↵
ถ้าเลือก U (Uniform points) จะมีพรอมต์ ดังนี้	
Alignment:<>	♦ ทิศทางของข้อความ ป้อนค่าโดยตรง หรือกำหนด 2 จุด
No. of Text:<>	♦ จำนวนข้อความที่จะเขียน
Distance:<>	♦ ระยะห่างของแต่ละข้อความ
Start point:	♦ จุดเริ่มต้น

1. พิจารณาการกำหนด limit ของกระดาษในภาวะ Paper space เป็นในหน่วยมิลลิเมตร และสัดส่วนที่ใช้ในการพิมพ์ออกทางเครื่องพิมพ์เป็น 1 Drawing unit = 1 มิลลิเมตร

2. ค่าป้อนเกี่ยวกับขนาดของรูปวาด หรือความสูงอักษรใด ๆ ที่เขียนบนภาวะ Paper space จะป้อนด้วยขนาดในหน่วยมิลลิเมตรที่ต้องการเมื่อพิมพ์ออกบนกระดาษ เช่น ต้องการเขียนอักษรขนาด 2 มม. ก็ป้อนด้วยค่า 2 มม.

ในข้อ 1 และ 2 ข้างต้นกล่าวสรุปสั้น ๆ คือในที่นี้เราจะถือว่าการทำงานทุกอย่างบนภาวะ Paper space จะเป็นหน่วยมิลลิเมตรเท่านั้น

3. ตัวคูณสำหรับการกำหนดขนาดที่จะป้อนในภาวะ Model space คำนวณได้จากผลหารของ DWGSC กับ TOMM โดย

DWGSC คือ ค่าสเกลของแบบ เช่น สเกล 1:100 ค่า DWGSC จะเท่ากับ 100

TOMM คือ ตัวคูณในการแปลงหน่วยเป็นหน่วยมิลลิเมตร ดังนั้นจะมีค่าเป็น 1 และ 1000 สำหรับแบบที่เขียนในหน่วยมิลลิเมตรและเมตรตามลำดับ

เช่น เขียนแบบบน Model space ในหน่วยเมตร ที่สเกล 1:100 ต้องการเขียนตัวอักษรให้มีขนาด 2 มิลลิเมตร เมื่อพิมพ์ออกกระดาษ ที่พรอมต์ป้อนค่าตัวอักษรต้องป้อนค่าเท่ากับ 0.2 (= 2x(100/1000)) แต่ถ้าเปลี่ยนเป็นเขียนในหน่วยมิลลิเมตร ให้ป้อนด้วยค่าเท่ากับ 200 (= 2x(100/1))

4. การพิมพ์แบบออกกระดาษจะกระทำที่ภาวะ Paper space โดยสร้างวิวขึ้นด้วยคำสั่ง Mview บนภาวะ Paper space และกำหนดภาวะเป็น Model space ด้วยคำสั่ง Mspace ทั้งที่อยู่บน Paper space (หรือที่ Tilemode= 0) ทำการซูมภาพใน Viewport ที่สร้างขึ้นด้วยคำสั่ง Zoom และป้อนค่า nXP โดย n เป็นตัวเลขจากผลหารของ TOMM กับ DWGSC (ความหมายของตัวแปรทั้งสองดูในข้อ 3) จะเห็นว่าค่านี้เป็นส่วนกลับของค่าในข้อ 3 จากนั้นรันคำสั่ง Pan เพื่อแพนภาพจนได้ตำแหน่งที่เหมาะสมแล้วเซตกลับภาวะ Paper space ด้วยคำสั่ง Pspace อีกครั้ง

เช่น จากตัวอย่างในข้อ 3 ค่า n แบบในหน่วยเมตรและมิลลิเมตรจะมีค่าเท่ากับ 10 (=1000/100) และ 0.01 (=1/100) ตามลำดับ

หมายเหตุ ตัวแปรระบบที่กำหนดภาวะ Paper Space และ Model Space คือ Tilemode โดยมีค่าเป็น 0 และ 1 ตามลำดับ ที่ Tilemode= 0 เรายังสามารถเปลี่ยนไปมาระหว่างการทำงานบน Paper space กับ Model space ด้วยคำสั่ง Pspace และ Mspace ตามลำดับ

การสร้างไฟล์โปรแกรม

ไฟล์โปรแกรม AutoLISP จะต้องสร้างด้วยโปรแกรม Text editor ที่สามารถบันทึก (save) ได้ในรหัสแอสกีตามที่ได้ยกตัวอย่างในหัวข้อสิ่งที่ต้องการพื้นฐาน สำหรับในที่นี้จะขอยกตัวอย่างวิธีการใช้งาน Edit.com ซึ่งมีมาใน MS-DOS ตั้งแต่เวอร์ชัน 5 ขึ้นไป ที่ดอสพรอมต์ดังนี้

C:\> พิมพ์ EDIT \LISP\TXINC.LSP ↵

ในตัวอย่างนี้เก็บไฟล์ไว้ที่ไดเรคทอรี ชื่อ LISP กรณีที่ต่างจากนี้ก็ให้ระบุตามนั้น จากนั้นจะมีการเข้าสู่หน้าจอของโปรแกรม Text editor สามารถใช้เขียนข้อความใด ๆ ได้ การป้อนข้อความใช้คีย์บอร์ดพิมพ์ได้โดยตรง การลบเมื่อใช้แป้น Backspace จะลบอักษรหนึ่งตัวทางซ้ายของเคอร์เซอร์ และการลบด้วยแป้น Delete จะลบอักษรหนึ่งตัวทางขวาของ

เคอร์เซอร์ การเว้นวรรคใช้แป้น Spacebar ห้ามใช้ Tab การเลื่อนไปตำแหน่งใด ๆ ของข้อความใช้แป้นลูกศร ขึ้น-ลง ซ้าย-ขวา หรือใช้เม้าส์ชี้เลือกในตำแหน่งที่ต้องการ และเมื่อป้อนข้อความจนครบแล้วให้ทำการบันทึก (save) โดยกด Alt จะมีแถบสี (Highlight) ที่บรรทัดเมนูบนสุด เลือกหมวด File กดแป้นลูกศรชี้ลง เลือกรายการ Save จากนั้นออกจากโปรแกรมด้วยการกด Alt เลือกหมวด File อีกครั้ง และเลือกรายการ Exit จะกลับมาที่ดอสพรอมต์ และเมื่อต้องการแก้ไขใด ๆ อีกให้เรียกตามบรรทัดข้างต้น

ขอให้ผู้อ่านสร้างไฟล์ TXINC.LSP ตามรูปที่ 8 ก่อน โดยมีข้อสังเกตดังนี้

ข้อสังเกต

1. ไฟล์ AutoLISP เป็นไฟล์รหัสแอสกีที่มีนามสกุลเป็น .LSP

2. บรรทัดหรือข้อความที่ขึ้นต้นด้วยเครื่องหมาย ; (semicolon) ใช้ในการบรรยาย (comment) โปรแกรมเท่านั้น ไม่ได้เป็นส่วนที่จะถูกสั่งงาน ผู้อ่านอาจจะละออกไม่ต้องพิมพ์ก็ได้

3. ให้ระมัดระวังเรื่องตัวสะกด อักษรโอ และเลขศูนย์ (o, 0) อักษรแอล และเลขหนึ่ง (l,1)

4. โดยทั่วไปโปรแกรม AutoLISP จะถือว่าอักษรตัวใหญ่และเล็กให้ผลเหมือนกัน ยกเว้นอักษร n ใน \n ต้องใช้ตัวพิมพ์เล็กเท่านั้น ห้ามใช้ตัวพิมพ์ใหญ่ และข้อความในฟังก์ชัน Princ ต้องป้อนตัวใหญ่และเล็กตามรูปร่างที่ต้องการให้แสดง

5. จำนวนเว้นวรรค จะมีจำนวนเท่าใดก็ได้ ในบริเวณที่มีแสดงเว้นวรรคไว้ ทั้งนี้อาจจะมีการขึ้นบรรทัดใหม่ต่างจากที่แสดงไว้ก็ได้ แต่ขอแนะนำให้สร้างเหมือนในบทความ เพื่อง่ายต่อการตรวจสอบความถูกต้อง

6. จำนวนวงเล็บเปิดต้องเท่ากับจำนวนวงเล็บปิด เครื่องหมายคำพูด ("") ต้องมีจำนวนเป็นคู่

การโหลดโปรแกรม AutoLISP

ก่อนใช้งานโปรแกรม AutoLISP จะต้องมีการโหลดไฟล์เข้าสู่หน่วยความจำของเครื่องก่อน โดยใช้ฟังก์ชัน LOAD ของ AutoLISP รันที่บรรทัด Command: ใน AutoCAD ในรูปดังนี้

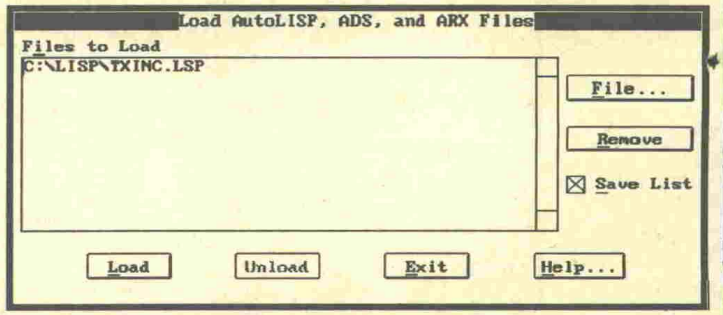
(LOAD ชื่อไฟล์ AutoLISP)

ชื่อไฟล์นี้ต้องระบุ path ด้วยโดยใช้เครื่องหมาย forward slash (/) ต่างจากในดอสที่ใช้ back slash (\) แต่ไม่ต้องระบุนามสกุล ตัวอย่างเช่น เก็บไฟล์ไว้ในไดเรคทอรี \LISP ให้ป้อนที่บรรทัด Command: ดังนี้

Command: พิมพ์ (LOAD "/LISP/TXINC") ↵
C:TXINC

เมื่อไม่มีข้อผิดพลาดอะไรในโปรแกรมที่สร้างขึ้น จะพบกับข้อความ C:ชื่อคำสั่งที่นิยามไว้ ในที่นี้คือ C:TXINC ถ้าต่างจากนี้ไปขอให้ผู้อ่านกลับไปแก้ไขไฟล์และบันทึก (save) แล้วทดสอบโหลดใหม่อีกครั้ง

นอกจากนั้นเราสามารถจะโหลดได้ง่ายขึ้นโดยการใช้คำสั่ง APPLOAD ซึ่งจะเป็นคำสั่งที่ใช้ในการโหลดแอพพลิเคชันในรูปไดอะล็อกบ็อกซ์ ซึ่งค่อนข้างง่ายต่อการใช้งาน ดูรูปที่ 6 ประกอบ



รูปที่ 6 ไดอะล็อกบ็อกซ์จากคำสั่ง APPLOAD ซึ่งใช้ในการโหลดไฟล์ AutoLISP ได้

การกำหนดให้โปรแกรมโหลดเอง

เพื่อให้สะดวกต่อการใช้งานคำสั่งจากโปรแกรม AutoLISP มากขึ้นอีก เราสามารถทำให้โปรแกรมมีการโหลดตัวเองได้โดยเราไม่ต้องมารันคำสั่ง LOAD หรือ APPLOAD เพื่อโหลดไฟล์เข้าหน่วยความจำก่อนการใช้งาน ในการนี้เราสามารถทำได้โดยอาศัยไฟล์ ACAD.LSP ซึ่งเป็นไฟล์รหัสแอสกีเช่นกัน มีข้อเด่นคือเป็นไฟล์ AutoLISP ที่จะถูกโหลดทุกครั้งที่มีการเปิดไฟล์แบบ (Open or New Drawing) ให้ทำได้จากเรียกไฟล์นี้ มาแก้ไข เพิ่มบรรทัดกำหนดการโหลดไฟล์ TXINC.LSP เข้าไว้ที่ท้ายไฟล์เดิมนี้ เช่น (LOAD "/LISP/TXINC") และในกรณีที่ยังไม่มีไฟล์ ACAD.LSP เลยเราก็สามารถสร้างใหม่ด้วย Text editor ใด ๆ โดยเก็บไว้ใน path ของโปรแกรม AutoCAD หรือในไดเรคทอรี SUPPORT เช่น \ACAD, \ACAD\SUPPORT, \ACADR13\COMMON\SUPPORT

ข้อสังเกตการใช้คำสั่ง TXINC

คำสั่ง TXINC การใช้งานค่อนข้างตรงไปตรงมา ตามความหมายของแต่ละพรอมต์ที่ได้ระบุไปแล้ว ในหัวข้อนี้เราจะกล่าวถึงข้อสังเกตทั่วไป ดังนี้

1. การเลือก Justify ซึ่งเป็นพรอมต์แรกของคำสั่งนี้ ต้องกำหนดให้เหมาะสมกับรูปวาดชนิดหนึ่ง ๆ เช่น ในการกรอกเลขหมายในวงกลม กำหนด Justify แบบ MC (Middel Center) จะได้ข้อความอยู่กึ่งกลางจุดศูนย์กลางพอดี ดูรูปที่ 1 และ 2

การกรอกค่าในกรอบสี่เหลี่ยมเช่นตาราง หรือในช่องขั้วต่อสาย (Terminal) ให้คำนึงถึงรูปแบบการชิด ขวา-ซ้าย หรือกึ่งกลางด้วย ดูผลในรูปที่ 5 จะเห็นว่า Justify ที่เหมาะสมจะเป็น ML และ MR เท่านั้น (กึ่งกลาง แบบชิดซ้าย หรือขวาตามลำดับ) ในการกำหนดจุดนั้นอาจต้องสร้างเส้นอ้างอิงไว้ก่อน แล้วค่อยลบออกเมื่อวาดเสร็จแล้ว

2. เมื่อต้องการยกเลิกการทำงานของคำสั่งนี้ ทำได้โดยใช้คำสั่ง U (Undo) ซึ่งจะให้ผลลบทุกข้อความที่เขียนจากคำสั่งนี้ในครั้งหนึ่ง ๆ ออกหมดทั้งชุดพร้อมกัน (ไม่ได้ย้อนกลับ ลบทีละตัว)

3. ที่พรอมต์ Prefix และ Suffix เมื่อมีค่าดีฟอลต์อยู่ การทำให้ว่าง (หรือไม่ต้องการให้มีข้อความนำหน้าและลงท้าย) โดยไม่ป้อนค่าใด ๆ จะกด Enter หรือ Space bar ไม่ได้ ให้กระทำโดยพิมพ์ตัวอักษร ~ (tilde) แล้วกด Enter คำสั่ง TXINC จะมอง ~ ว่าเป็นการกำหนดค่าว่าง ๆ ให้กับพรอมต์นี้

4. ที่พรอมต์ First value ต้องไม่มีค่าเป็นแบบ Null value หรือกด Enter ผ่านโดยไม่มีค่าใด ๆ ทั้งที่เป็นค่าดีฟอลต์และการป้อนค่าจากพรอมต์เอง ถ้าป้อนแบบ Null value นี้จะมีข้อความเตือนว่า Now not a default value. Must be input value, first. และวนกลับมาที่พรอมต์เดิมให้ป้อนค่าใหม่ เป็นเช่นนี้จนกว่าจะได้ค่าแบบไม่เป็น Null value และเช่นกันที่พรอมต์รับค่าอื่น ๆ ของคำสั่งนี้ก็จะเป็นเช่นนี้เสมอ ยกเว้น Prefix และ Suffix ตามที่ระบุในข้อ 3

5. ที่พรอมต์ First value จะมีการกรอง (Filter) ไม่รับค่าที่อักขระตัวแรกไม่เป็นตัวอักษรหรือตัวเลข กรณีที่ป้อนค่าเช่นนี้จะได้รับข้อความเตือนว่า Value must be any numeric or A-Z letters. Try again!

สำหรับกรณีตัวแรกเป็นตัวอักษร ซึ่งจะเป็นได้ทั้ง a-z และ A-Z คำสั่ง TXINC จะใช้เฉพาะอักษรตัวแรกเท่านั้น เช่น ป้อนด้วยคำว่า ANT หรือ ant จะถูกกรองและกำหนดค่าเป็น A เท่านั้น ส่วนกรณีป้อนค่าแบบตัวเลขจะรับค่าไว้ทั้งหมด เช่น ป้อนด้วย 100 ก็จะถูกกำหนดค่าเป็น 100

อธิบายโปรแกรม

ในโปรแกรม TXINC นี้ประกอบด้วย ส่วนเมนโปรแกรม และ 7 สับรูทีน (ในการบรรยายจะใช้คำว่า สับรูทีน สำหรับฟังก์ชันที่สร้างขึ้นใหม่ ส่วนที่มีอยู่แล้วใน AutoLISP จะเรียกว่า ฟังก์ชัน) ดังนี้

บรรทัด 1-10 เป็นบรรทัด Comments ใช้บรรยายประกอบโปรแกรม ไม่ได้เป็นส่วนที่ใช้ทำงานของโปรแกรมแต่อย่างใด การทำ Comments ให้เริ่มต้นข้อความนั้น ๆ ด้วยเครื่องหมาย ; (semicolon) สำหรับในไฟล์ TXINC.LSP นี้ มีการใช้ ; มากกว่าหนึ่งตัวเป็น ;; หรือ ;;; นั้น ไม่ได้มีความหมายอะไรเป็นพิเศษ ใช้เพียงเพื่อจัดส่วนของโปรแกรมแยกให้เห็นชัดเจนขึ้นเท่านั้น

ตารางที่ 2 แสดงการใช้งานสับรูทีนในโปรแกรม TXINC

สับรูทีน	หน้าที่
(gsxdef prmpt gvar)	ใช้แสดงพรอมต์เพื่อรับค่าแบบสตริงหรือชุดตัวอักขระ โดยแสดงค่าดีฟอลต์ได้ (ค่าในวงเล็บ <>) และตรวจสอบไม่ให้ป้อนค่าแบบ Null value ด้วย prmpt และ gvar เป็นพรอมต์ที่จะแสดง และชื่อตัวแปรโกลบอล ในการใช้งานจะต้องกำหนด (declare) ค่าเริ่มต้นให้กับตัวแปรนี้ก่อน เป็น "" ตัวอย่างการใช้งาน เช่น ในบรรทัด 154 (gsxdef "First value" #txt0) ขอให้สังเกตว่ามีการกำหนดค่าเริ่มต้นให้กับตัวแปร #txt0 ในบรรทัดที่ 128 แล้ว ด้วยฟังก์ชัน setq
(gddef prmpt gvar)	ใช้ในแบบเดียวกับสับรูทีน gsxdef แต่รับค่าแบบระยะทางกำหนดค่าด้วยสองจุดใด ๆ หรือป้อนด้วยตัวเลขโดยตรง
(gidef prmpt gvar)	ใช้ในแบบเดียวกับสับรูทีน gsxdef แต่รับค่าแบบจำนวนเต็ม (ไม่มีจุดทศนิยม)
(gsdef prmpt gvar)	รับค่าแบบสตริง เช่นเดียวกับสับรูทีน gsxdef แต่ต่างกันตรงที่สับรูทีน gsdef ยอมรับค่าแบบ Null value ในการกด Enter ผ่าน โดยไม่ป้อนค่าใด ๆ เลย แม้แต่ในกรณีที่ค่าดีฟอลต์ยังไม่มีค่าก็ตาม
(txrng input)	ใช้ตรวจสอบว่า ค่า input เป็นตัวเลขหรือตัวอักษรหรือไม่ เพื่อกรองการป้อนค่าเป็นอักขระพิเศษอื่น ๆ โดยใช้ความรู้เกี่ยวกับค่ารหัสแอสกีที่ว่า เลข 0-9 มีรหัสแอสกีเป็น 48-57 อักษร a-z มีรหัสแอสกีเป็น 65-90 อักษร A-Z มีรหัสแอสกีเป็น 97-122 ถ้าเป็นจริง ให้ผลลัพธ์เป็นค่า input นั้น มิฉะนั้นให้ผลลัพธ์เป็น nil
(txincr type index)	ใช้กำหนดวิธีการอ่านค่า ถ้า type เป็นตัวอักษร (Alpha) ให้อ่านค่าด้วยฟังก์ชัน chr แต่ถ้าเป็นตัวเลข (Numeric) ให้อ่านด้วยฟังก์ชัน itoa
(godef prmpt gvar)	ใช้ในแบบเดียวกับสับรูทีน gsxdef แต่รับค่าแบบมุมซึ่งจะกำหนดค่าด้วยสองจุดใด ๆ หรือป้อนด้วยตัวเลขโดยตรง

บรรทัด 7-125 เป็นส่วนของสับรูทีนที่ใช้ทั้งหมด ขอให้สังเกตความหมายได้จากตารางที่ 1

บรรทัด 127 บรรทัด Comments

บรรทัด 128 ใช้กำหนดค่าเริ่มต้นของตัวแปรโกลบอลที่จะใช้กับสับรูทีน GXXDEF ทั้งหลาย

บรรทัด 130 บรรทัด Comments

บรรทัด 131-214 เป็นส่วนของเมนโปรแกรมที่ใช้นิยามคำสั่ง TXINC ขึ้นใหม่ และกำหนดขั้นตอนการทำงานให้แสดงพรอมต์เพื่อให้ผู้ใช้ป้อนค่าต่าง ๆ และนำมาจัดระเบียบข้อมูล และคำนวณหาชุดอักษร หรือตัวเลขในลำดับต่อไป รวมถึงกำหนดจุดของการเขียนด้วย ในกรณีที่กำหนดแบบสม่ำเสมอ

บรรทัด 131-133 นิยามคำสั่ง TXINC ด้วยฟังก์ชัน defun เริ่มต้นด้วยวงเล็บเปิด และมีวงเล็บปิดของฟังก์ชันนี้ ที่บรรทัด 214 กำหนดชื่อด้วย C:TXINC อักษร C: ไม่ได้ระบุถึงไดรฟ์ C แต่หมายถึงนิยามแบบให้เป็นคำสั่ง (Command) ใช้งานได้โดยตรงแบบคำสั่งปกติใน AutoCAD เอง วงเล็บถัดมาเป็นที่กำหนดตัวแปรแบบโลคอล (local) ซึ่งจะมีการ Reset ค่าตัวเองให้หายไปเมื่อจบการทำงานของคำสั่ง ตัวแปรใดที่ไม่ได้กำหนดอยู่ในวงเล็บนี้จะยังคงมีค่าอยู่ (เมื่อยังไม่มีการโหลดโปรแกรมใหม่) ซึ่งเรียกตัวแปรชนิดหลังนี้ว่าตัวแปรแบบโกลบอล (global)

หมายเหตุ

ตัวแปรแบบโกลบอลมีประโยชน์ทำให้เรากำหนดค่าดีฟอลต์เป็นค่าสุดท้ายที่เพิ่งป้อนไว้จากการรันคำสั่งได้ ในคำสั่งนี้จะเลือกตั้งชื่อตัวแปรชนิดนี้ให้นำหน้าด้วยเครื่องหมาย #

บรรทัด 134 อ่านค่าเบื้องต้นของระบบเก็บไว้ก่อน ดังนี้

ตัวแปร olderr เก็บฟังก์ชัน *error* , ตัวแปร oldcmd เก็บค่าตัวแปรระบบ cmdecho และตัวแปร oldblp เก็บค่าตัวแปรระบบ blipmode

การที่เราอ่านค่าเหล่านี้มาเก็บไว้ก่อนเนื่องจากเราจะมีการเปลี่ยนแปลงค่าเหล่านี้ให้เหมาะสมกับการทำงานของโปรแกรม TXINC และเพื่อไม่ให้กระทบกับค่าเดิมของระบบ เมื่อจบการทำงานของโปรแกรมจะต้องมีการกำหนดค่าเก่าคืนอย่างถูกต้อง จึงต้องอ่านเก็บไว้อ้างอิง จุดที่เราจะกำหนดค่าคืนมีสองตำแหน่ง คือ

1. เมื่อโปรแกรมถูกสั่งยกเลิก (Cancel) ขณะกำลังทำงานด้วยการกด ESC หรือ Ctrl+C ซึ่งควบคุมด้วยฟังก์ชัน *error* ที่เรานิยามใหม่ในบรรทัด 136-145 ให้ดูการอธิบายฟังก์ชัน *error* ประกอบ

2. เมื่อกระทำทุกขั้นตอนของโปรแกรมจากการรันคำสั่งแบบปกติ จบลงที่บรรทัด 210-212

หมายเหตุ

1. การกำหนดค่าตัวแปรทั่ว ๆ ไป ในการโปรแกรม AutoLISP ใช้ฟังก์ชัน setq ในรูป

(setq ชื่อตัวแปร ค่าที่จะเก็บ)

เช่น (setq olderr *error*) จะเก็บค่าเป็นฟังก์ชัน *error* ไว้ที่ตัวแปร olderr,

(setq a 1.0) จะเก็บค่า 1.0 ไว้ที่ตัวแปร a

2. ส่วนการกำหนดค่าตัวแปรระบบ ใช้ฟังก์ชัน setvar ในรูป (setvar ชื่อตัวแปรระบบ ค่าที่จะเก็บ)

เช่น (setvar "cmdecho" 0) จะกำหนดค่าตัวแปรระบบ cmdecho เป็น 0

3. การอ่านค่าจากตัวแปรทั่ว ๆ ไป ในการโปรแกรม AutoLISP จะใช้การกำหนดชื่อตัวแปรนั้น ๆ โดยตรง

เช่น (setq *error* olderr) จะเป็นการอ่านค่าที่เก็บไว้ในตัวแปร olderr มาเก็บไว้ที่ตัวแปร *error* ซึ่งเป็นฟังก์ชันหนึ่ง

ตารางที่ 3 แสดงฟังก์ชัน AutoLISP ที่ใช้ในคำสั่ง TXINC

ฟังก์ชัน	หน้าที่
กลุ่มตรวจสอบเงื่อนไข	
if	ตรวจสอบเงื่อนไขหนึ่งแล้วแยกวิธีการทำงานต่อมา
cond	คล้าย ๆ if แต่มากกว่าหนึ่งเงื่อนไข
กลุ่มควบคุมให้กระทำซ้ำ	
while	ควบคุมให้กระทำซ้ำเมื่อเงื่อนไขที่กำหนดยังคงเป็นจริง หรือมีค่าอยู่ ไม่เป็น Null value
repeat	ควบคุมให้กระทำซ้ำตามจำนวนรอบที่กำหนด
กลุ่มรับค่าป้อนจากผู้ใช้	
getdist	แสดงพรอมต์รับค่าแบบระยะทางกำหนดโดยสองจุด หรือป้อนตัวเลขโดยตรง

ตารางที่ 3 (ต่อ)

ฟังก์ชัน	หน้าที่
getint	แสดงพรอมต์รับค่าแบบจำนวนเต็ม
getkword	แสดงพรอมต์รับค่าแบบคำ หรือตัวอักษร
getorient	แสดงพรอมต์รับค่าแบบมุมกำหนดโดยสองจุด หรือป้อนตัวเลขโดยตรง
getpoint	แสดงพรอมต์รับค่าแบบจุด
getstring	แสดงพรอมต์รับค่าแบบสตริง หรือข้อความอักษร
initget	ใช้ควบคุมกำหนดลักษณะการทำงานของกลุ่มคำสั่ง getxxx ทั้งหลาย
กลุ่มกำหนดค่า	
setq	ใช้กำหนดค่าให้กับตัวแปรใด ๆ
setvar	ใช้กำหนดค่าให้กับตัวแปรระบบ
กลุ่มจัดการสตริง หรือตัวอักษร	
strcase	ใช้แปลง Case หรือการเป็นตัวพิมพ์เล็กหรือใหญ่ของตัวอักษร
strcat	ใช้ต่อข้อความสตริงมากกว่าหนึ่งเข้าด้วยกันเป็นสตริงเดียว
กลุ่มตรวจสอบตรรกะ	
=	เท่ากับ
/=	ไม่เท่ากับ
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ
or	หรือ
and	และ
not	ไม่ เหมือนกำหนดเป็นนิเสธ (negation)
กลุ่มแปลงค่า	
angtos	แปลงค่าระหว่างระบบของมุม ในที่นี้ใช้แปลงจากเรเดียนเป็นองศา
ascii	แปลงค่าสตริงเป็นรหัสแอสกีที่สมมูล เช่น (ascii "A") เป็น 65
chr	แปลงค่าตัวเลขเป็นสตริงที่สมมูล โดยมองตัวเลขเป็นรหัสแอสกี เช่น (chr 65) เป็น "A"
atoi	แปลงค่าสตริงเป็นจำนวนเต็ม เช่น (atoi "123") เป็น 123
itoa	แปลงจำนวนเต็มเป็นสตริง เช่น (itoa 123) เป็น "123"
อื่น ๆ	
defun	นิยามคำสั่ง หรือสับรูทีน
command	ใช้คำสั่งใน AutoCAD
princ	- เมื่อใช้ตามด้วยข้อความเป็นการแสดงพรอมต์ที่บรรทัด Command: เช่น (princ "\n Hello.") แสดงข้อความที่
	บรรทัด Command: เป็น "Hello"
	- ใช้แบบไม่มีข้อความตาม ในรูป (princ) เป็นการกำจัดการแสดงผลลัพธ์สุดท้ายจากการทำงานของโปรแกรม
member	ใช้ตรวจสอบว่าเป็นสมาชิกในวงเล็บ หรือ List หรือไม่
getvar	ใช้ในการอ่านค่าตัวแปรระบบ

(ascii #txt0) เป็นการกำหนดให้ใช้ฟังก์ชัน ascii กับค่าที่เก็บไว้ในตัวแปร #txt0 (กล่าวคือมันถูกอ่านค่าได้โดยตรง เพียงระบุชื่อตัวแปรให้ตรงเท่านั้น) สำหรับหน้าที่ของฟังก์ชัน ascii ดูในตารางที่ 3

4. การอ่านค่าจากตัวแปรระบบ จะต่างจากการอ่านค่าตัวแปรทั่ว ๆ ไป ไม่สามารถระบุเพียงชื่อตัวแปรเท่านั้น ต้องใช้ฟังก์ชัน getvar ด้วย

เช่น (setq oldblp (getvar "blipmode")) จะเป็นการอ่านค่าตัวแปรระบบ blipmode และ นำมาเก็บค่าไว้ที่ตัวแปร oldblp

บรรทัด 135 ปิดการแสดงพรอมต์ใด ๆ ของคำสั่งปกติใน AutoCAD ซึ่งทำได้โดยกำหนดตัวแปรระบบ cmdecho ให้มีค่าเป็น 0 (ส่วนการกำหนดให้เป็นภาวะเปิด ให้แสดงค่า จะใช้ค่าเป็น 1)

บรรทัด 136-145 ส่วนนี้เป็นการนิยามฟังก์ชัน *error* ขึ้นใหม่ ฟังก์ชัน *error* ปกติมีอยู่แล้วใน AutoCAD เอง ซึ่งมีการทำงานปกติเป็นตัวตรวจจับความผิดพลาดที่เกิดขึ้นจากการทำงานของโปรแกรม AutoLISP โดยจะกำหนดให้มีการแสดง code ของโปรแกรมในส่วนที่ทำให้การทำงานผิดพลาดออกมาที่บรรทัด Command: ซึ่งในส่วนนี้จะมีประโยชน์ระหว่างที่เรากำลังพัฒนาโปรแกรม ช่วยให้ทราบว่าผิดพลาดที่ใด จะได้แก้ไขได้ (debug) อย่างไรก็ตาม การสั่งยกเลิก (Cancel) ด้วยการกด Ctrl+C หรือ ESC จัดเป็นความผิดพลาดของการทำงานด้วย ดังนั้นเมื่อสั่งยกเลิกการทำงานของคำสั่งที่พัฒนาจากการโปรแกรม AutoLISP นี้ ก็จะมีการแสดง Codes ออกมา บางครั้งอาจเป็นร้อยบรรทัด ซึ่งต่างจากการใช้คำสั่งปกติใน AutoCAD ที่เมื่อสั่งยกเลิกจะไม่แสดงอะไรเลย นอกจากคำว่า *Cancel* เพื่อให้การทำงานเป็นไปในลักษณะเดียวกันจึงต้องมีการนิยามหน้าที่ให้ฟังก์ชันนี้ใหม่

ในคำสั่ง TXINC นิยามให้ฟังก์ชัน *error* ทำงานกำหนดค่าคืนระบบเดิม ซึ่งได้แก่ค่าตัวแปรระบบที่เราได้เปลี่ยนค่าไป ได้แก่ cmdecho และ blipmode รวมถึงตัวฟังก์ชัน *error* ด้วย นอกจากนั้นยังมีกำหนดจุดจบของการกำหนดตำแหน่งของคำสั่ง U ด้วยการใช้ฟังก์ชัน command กับคำสั่ง UNDO

บรรทัด 146 และ 209 สองบรรทัดนี้เป็นกุญแจหลักที่ทำให้เมื่อรันคำสั่ง U จะยกเลิกการทำงานของคำสั่ง TXINC ได้ทั้งหมดพร้อมกันในการสั่ง U ครั้งเดียว ทุกข้อความที่ถูกเขียนขึ้นจากคำสั่ง TXINC ในคราวหนึ่ง ๆ จะหายไปพร้อม ๆ กัน ไม่ต้องสั่ง U ทีละครั้งต่อแต่ละข้อความ ในการนี้ใช้ฟังก์ชัน command กับคำสั่ง UNDO ในตัวเลือก begin และ end โดย begin ใช้กำหนดก่อนบรรทัดที่จะนับเป็นการกระทำแรกที่จะยกเลิก และใช้ end ในหลังบรรทัดสุดท้ายที่จะยกเลิก หรือก็คือการกำหนดกลุ่มของการกระทำที่จะยกเลิกนั่นเอง

นอกจากนี้ในบรรทัด 140 ส่วนของการนิยามฟังก์ชัน *error* ก็ใช้ฟังก์ชัน command กับคำสั่ง UNDO เช่นกัน ใช้กำหนดปิดกลุ่มของการกระทำของคำสั่ง TXINC ที่จะถูกยกเลิกด้วยคำสั่ง U ในกรณีที่มีการยกเลิก จากการกด Ctrl+C หรือ ESC

สำหรับผู้ใช้รีลีส 12 ขอให้ใช้ตัวเลือก group แทน begin ในบรรทัดที่ 146 นี้

บรรทัด 147-150 ใช้แสดงพรอมต์ Center/Middle/Right/TL/TC/TR/ML/MC/MR/BL/BC/BR: โดยใช้ฟังก์ชัน initget ร่วมกับ getkword

ฟังก์ชัน initget ใช้ในการควบคุมการป้อนค่าที่พรอมต์ของฟังก์ชัน getxxx ทั้งหลาย (getkword, getstring, getdist, ...) มีรูปแบบเป็น (initget bit keyword) สำหรับในโปรแกรมนี้ใช้เฉพาะกรณีค่า bit (บิต) ถูกกำหนดเป็น 1 ซึ่งหมายถึงต้องป้อนค่าแบบไม่เป็น Null value หรือต้องมีค่าเท่านั้น ส่วนคีย์เวิร์ด (keyword) จะใช้กับฟังก์ชัน getkword บังคับให้คำที่ป้อนต้องเป็นไปตามที่กำหนดไว้ในคีย์เวิร์ดเท่านั้น

ในการนี้เราให้ป้อนค่าเพื่อกำหนด Justify ของตัวอักษร เฉพาะค่าตามที่กำหนดไว้เป็นคีย์เวิร์ด และเก็บไว้ที่ตัวแปร justify

บรรทัด 151 ใช้แก้ปัญหาเมื่อผู้ใช้ตอบ justify แบบ Center คำสั่ง TEXT จะมองว่าเป็น Object snap แบบ CENTER จึงใช้ฟังก์ชัน if มีเงื่อนไขตรวจสอบว่า ถ้า justify เป็น Center ให้กำหนดค่าใหม่เป็น C แทน

บรรทัด 152 กำหนดค่าเริ่มต้นให้กับตัวแปร cont เป็นค่าคงที่ T

บรรทัด 153-159 เป็นกลุ่มนิพจน์ที่ใช้แสดงพรอมต์ First value: โดยใช้ฟังก์ชัน while บังคับให้วนแสดงพรอมต์นี้จนกระทั่งตัวแปร cont มีค่าเป็น nil สำหรับตัวแปรนี้จะถูกเปลี่ยนแปลงค่าได้จากการใช้ฟังก์ชัน if ที่มีเงื่อนไขตรวจสอบว่า ค่าที่ป้อนเป็นตัวเลข หรือตัวอักษร หรือไม่ ด้วยการใช้สับรูทีน txrng (ดูนิยามของ txrng ในตารางที่ 2) ถ้าใช่ให้ออกจาก loop ของ while นี้ได้ มิฉะนั้นให้แสดงข้อความ Value must be any numeric or A-Z letter. Try again! และวนกลับไปที่พรอมต์ First value: ใหม่อีกครั้ง

มีจุดสังเกตอีกอย่างหนึ่งคือ ไม่ว่าค่าที่ป้อนเข้ามาจะเป็นอักษรตัวพิมพ์ใหญ่หรือเล็ก จะถูกแปลงเป็นแบบตัวใหญ่ทั้งสิ้น ด้วยฟังก์ชัน strcase ก่อนที่จะกำหนดค่าให้กับตัวแปร #txt0

บรรทัด 160 แปลงค่าที่ป้อนเข้ามาที่พรอมต์ First value: ให้เป็นรหัสแอสกี (เช่น อักษร A มีรหัสแอสกีเป็น 65)

บรรทัด 161-167 กำหนดค่าเริ่มต้นของการเขียนไว้ที่ตัวแปร txt โดยแยกเงื่อนไขด้วยฟังก์ชัน cond ดังนี้

ถ้าเป็นตัวเลข หรือมีรหัสแอสกีในช่วง 48-57 ให้กำหนดค่าตัวแปร txt เก็บค่าเป็นตัวเลขที่สมมูลกับ ค่าที่ป้อน และมีค่าน้อยกว่าอยู่หนึ่ง เนื่องจากค่าที่ป้อนที่พรอมต์ First value: จะเป็นแบบสตริง จึงใช้ฟังก์ชัน atoi แปลงเป็นตัวเลข นอกจากนั้นกำหนดประเภทข้อมูลไว้ด้วย เป็น Numeric

ถ้าเป็นตัวอักษร ให้กำหนดค่าตัวแปร txt เก็บค่าเป็นตัวเลขที่สมมูลกับค่าที่ป้อน และมีค่าน้อยกว่าอยู่หนึ่ง เนื่องจากค่าที่ป้อนที่พรอมต์ First value: จะเป็นแบบสตริง จึงใช้ฟังก์ชัน ascii แปลงเป็นตัวเลขในรูปรหัสแอสกี (เป็นจำนวนเต็ม) นอกจากนั้นกำหนดประเภทข้อมูลไว้ด้วย เป็น Alpha

บรรทัด 168-170 แสดงพรอมต์ Prefix: และ Suffix: เพื่อรับข้อความที่จะใช้นำหน้าหรือลงท้ายตามลำดับ โดยเก็บไว้ที่ตัวแปร #prefix และ #suffix

บรรทัด 171-172 กรณีรันคำสั่ง TXINC ซ้ำ (โดยยังไม่มีการโหลด DWG ใหม่) จะมีค่าดีฟอลต์ที่พรอมต์ Prefix: และ Suffix: ตามค่าเดิมของการรันครั้งก่อน และถ้าเป็นแบบไม่ว่างเปล่า (Not null value) จะทำให้ไม่สามารถป้อนค่าใหม่แบบ null value ได้ เพื่อให้มีการใช้งานคำสั่งนี้แบบไม่มี Prefix และ Suffix ได้อีก ซึ่งการกด Space bar ก็จะไม่ให้ผลตามที่เราต้องการ ในคำสั่ง TXINC จึงกำหนดให้ใช้การป้อนอักขระ ~ (tilde) เป็นค่าพิเศษ

ในบรรทัดทั้งสองนี้ใช้ฟังก์ชัน if โดยมีเงื่อนไขตรวจสอบว่าค่าที่ป้อนในพรอมต์ Prefix: และ Suffix: เป็น ~ หรือไม่ ถ้าใช่ให้กำหนดค่าเป็น "" ทำให้แก้ปัญหาป้อนค่าว่างเปล่าได้

บรรทัด 173 แสดงพรอมต์ Height: เพื่อรับความสูงที่จะใช้เขียนข้อความ

บรรทัด 174-175 คล้าย ๆ กับบรรทัด 147-150 ใช้ฟังก์ชัน initget ร่วมกับ getkword แต่ในที่นี้สำหรับการกำหนดรูปแบบตำแหน่งของการเขียนว่าเป็นแบบ Uniform หรือ Random

บรรทัด 176-208 เป็นส่วนควบคุมการเขียนข้อความ โดยใช้ฟังก์ชัน cond แยกเงื่อนไขเป็น 2 กรณี ดังนี้

กรณีที่ 1 แบบ Uniform

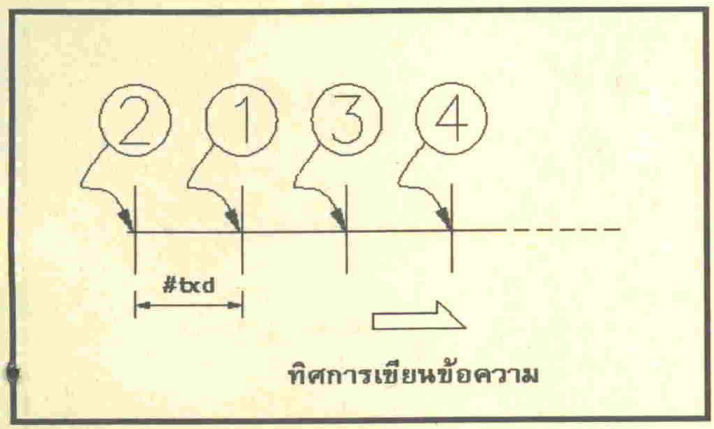
บรรทัด 178 แสดงพรอมต์ Alignment:

บรรทัด 179 แสดงพรอมต์ No. of Text:

บรรทัด 180 แสดงพรอมต์ Distance:

บรรทัด 181 กำหนด initget แบบบิตเป็น 1 เพื่อบังคับให้ฟังก์ชัน getpoint ในบรรทัดถัดมา ไม่รับค่าแบบ Null value

บรรทัด 182 แสดงพรอมต์ Start point:



รูปที่ 7 แสดงการชดเชยตำแหน่งของการเขียน

บรรทัด 183 กำหนดตัวแปรระบบ blipmode เป็นภาวะปิด หรือค่าเป็นศูนย์ เพื่อไม่ให้แสดงรอยกากบาท (blip) เมื่อมีการเลือกจุดบนหน้าจอ ซึ่งในกรณี Uniform โปรแกรมจะเลือกจุดการเขียนเอง (ไม่ได้กระทำโดยผู้ใช้) จึงไม่มีความจำเป็นใด ๆ ที่จะแสดงรอยกากบาทให้สังเกตเห็นตำแหน่งจุดที่เลือก

บรรทัด 184 กำหนดจุด Start point: ไว้เป็นจุดสุดท้ายที่มีการเลือก โดยใช้ตัวแปรระบบ lastpoint

บรรทัด 185 ชดเชย (compensate) ตำแหน่งของการเขียน ดูรูปที่ 7 ประกอบ

เนื่องจากการเขียนข้อความทั้งหมดนี้จะใช้นิพจน์การเขียนเดียวกัน คือในบรรทัด 186-194 จึงต้องมีการชดเชยค่าแรกที่ใช้กำหนดตำแหน่งก่อน เพื่อให้เข้าใจได้ชัดเจน ขอยกตัวอย่างกรณีเขียนข้อความเรียงไปในทิศตามรูปที่ 7 จากจุดที่ 1 ที่เลือกกำหนดไว้ที่พรอมต์ Start point: จะชดเชยค่าไปไว้ที่จุดที่ 2 โดยใช้นิพจน์

```
(strcat "@" (rtos (- #txd) 2 8) "<" #txalgn)
```

เช่น เมื่อระยะห่างการเขียนหรือ #txd เท่ากับ 4 หน่วย และทิศการเขียนหรือ #txalgn เท่ากับ 0 องศา จะได้ว่า

(strcat "@" (rtos (- 4) 2 8) "<" "0") จะเท่ากับ "@-4.00000000<0"

เมื่อถูกกำหนดตำแหน่งเขียนในบรรทัด 188 ค่าจะถูกกำหนดกลับไปที่ตำแหน่งที่ 1 ด้วยนิพจน์

(strcat "@" (rtos 4 2 8) "<" "0") จะเท่ากับ "@4.00000000<0"

และเมื่อเขียนจุดถัดมาเป็นจุดที่ 3, 4, ... จะเป็น "@4.00000000<0" เทียบกับจุดที่ถูกเขียนก่อนหน้า

บรรทัด 186-194 ใช้ควบคุมให้เขียนซ้ำจนครบตามข้อมูลที่กำหนดไว้ด้วยฟังก์ชัน repeat กระทำซ้ำเท่ากับจำนวนอักษรที่จะเขียนที่กำหนดไว้ในตัวแปร #txn ในการเขียนนี้ใช้ฟังก์ชัน command ให้รันคำสั่ง TEXT เพื่อเขียน โดยกำหนด Justify ตามค่าที่กำหนดไว้ในตัวแปรนี้ สำหรับบรรทัด 188 ดูความหมายในข้างต้น

บรรทัด 189 #txh และ 0.0 เป็นค่าความสูงอักษรและมุมหมุนตามลำดับ

บรรทัด 190 เป็นการกำหนดข้อความที่จะเขียน กำหนดด้วยนิพจน์

```
(strcat #prefix (txincr txt0 (setq txt (1+ txt))) #suffix)
```

ฟังก์ชัน strcat ใช้ในการนำข้อความมาต่อกัน ในที่นี้มี 3 ส่วน คือ #prefix, ชุดอักษรหรือเลขเรียง และ #suffix ขอให้สังเกตว่านิพจน์ที่กำหนดชุดอักษรหรือเลขเรียงนี้มีการเพิ่มค่าขึ้นทีละหนึ่งด้วย (1+ txt) ส่วนตัวควบคุมการแปลงค่าคือสับรูทีน txincr โดยมีค่า txt0 ระบุประเภทว่าเป็นอักษร (Alpha) หรือตัวเลข (Numeric) ดูบรรทัดที่ 163 และ 165

กรณีที่ 2 แบบ Random

บรรทัด 197 กำหนดตัวแปรระบบ blipmode เป็นภาวะเปิด หรือค่าเป็นหนึ่ง เพื่อให้แสดงรอยกากบาท (blip) เมื่อมีการเลือกจุดบนหน้าจอ ซึ่งในกรณี Random ผู้ใช้จะเลือกจุดการเขียนเอง จึงต้องมีรอยกากบาทให้สังเกตเห็นตำแหน่งจุดที่เลือก

บรรทัด 198-206 ควบคุมให้วนมาที่พรอมต์ Draw point: เพื่อให้ผู้ใช้เลือกจุดที่จะเขียนจนกว่าจะมีการกด Enter ผ่านไม่มีจุดที่เลือก ในการนี้กระทำโดยใช้ฟังก์ชัน while มีเงื่อนไขกำหนดให้วนกระทำเป็นการเลือกจุดโดยใช้ฟังก์ชัน getpoint

สำหรับในส่วนการของการเขียนใช้ฟังก์ชัน command และคำสั่ง TEXT การกำหนดขั้นตอนการทำงานจะเหมือนกับในแบบ Uniform ยกเว้นค่าจุดที่ป้อน จะนำค่าจากการป้อนของผู้ใช้

บรรทัด 213

ใช้ฟังก์ชัน princ แบบไม่มีข้อความใด ๆ ตาม เป็น (princ) เพื่อกำจัดการแสดงผลลัพธ์ตัวสุดท้ายที่บรรทัด Command:

รูปที่ 8 ไฟล์โปรแกรม TXTINC.LSP

```
1.    ;;; Draw Incrementing Text.
2.    ;;; Can specified Prefix and Suffix.
3.    ;;; 2 Methods in lettering.
4.    ;;; Uniform distance or Random.
5.    ;;; Developed by S.Chatchawal, 1995.
6.
7.    ;; GSXDEF (GetString with DEFault value check subroutine)
8.    ;; Not null value.
9.    ;; Argument: a prompt string & a global variabe name.
10.   ;; Must be declare a global variable,gvar. Before use GSXDEF.
11.   (defun gsxdef ( prmpt gvar / bye var)
12.      (setq bye "N")
13.      (while (/= bye "Y")
14.        (princ (strcat "\n  " prmpt ":<"))
15.        (princ gvar)
16.        (princ "> ")
17.        (setq var (getstring))
18.        (if (= var "")
19.            (if (/= gvar "")
20.                (setq bye "Y")
21.                (princ "\n!!! Now not a default value. Must be input value, first.")
22.            )
23.            (setq bye "Y" gvar var)
24.        )
25.      )
26.      gvar
27.   )
28.
29.   ;; GDDEF (GetDist with DEFault value check subroutine)
30.   ;; Argument: a prompt string & a global variabe name.
31.   ;; Must be declare a global variable,gvar. Before use GDDEF.
32.   (defun gddef ( prmpt gvar / bye var)
33.      (setq bye "N")
34.      (while (/= bye "Y")
35.        (princ (strcat "\n  " prmpt ":<"))
36.        (princ gvar)
37.        (princ "> ")
38.        (setq var (getdist))
39.        (if (= var nil)
40.            (if (/= gvar "")
41.                (setq bye "Y")
42.                (princ "\n!!! Now not a default value. Must be input value, first.")
43.            )
44.            (setq bye "Y" gvar var)
45.        )
46.      )
47.      gvar
```

รูปที่ 8 ไฟล์โปรแกรม TXTINC.LSP

```
48.   )
49.
50.   ;; GIDEF (GetInt with DEFault value check subroutine)
51.   ;; Argument: a prompt string & a global variabe name.
52.   ;; Must be declare a global variable,gvar. Before use GIDEF.
53.   (defun gidef ( prmpt gvar / bye var)
54.      (setq bye "N")
55.      (while (/= bye "Y")
56.        (princ (strcat "\n  " prmpt ":<"))
57.        (princ gvar)
58.        (princ "> ")
59.        (setq var (getint))
60.        (if (= var nil)
61.            (if (/= gvar "")
62.                (setq bye "Y")
63.                (princ "\n!!! Now not a default value. Must be input value, first.")
64.            )
65.            (setq bye "Y" gvar var)
66.        )
67.      )
68.      gvar
69.   )
70.
71.   ;; GSDEF (GetString with DEFault value check subroutine)
72.   ;; Argument: a prompt string & a global variabe name.
73.   ;; Must be declare a global variable,gvar. Before use GSDEF.
74.   (defun gsdef ( prmpt gvar / var)
75.        (princ (strcat "\n  " prmpt ":<"))
76.        (princ gvar)
77.        (princ "> ")
78.        (setq var (getstring))
79.        (if (/= var "") (setq gvar var))
80.        gvar
81.   )
82.
83.   ;; TXRNG
84.   ;; 0-9, a-z and A-Z range.
85.   (defun txrng ( input / chk )
86.      (setq chk (ascii input))
87.      (if (or
88.              (and (>= chk 48)(<= chk 57))
89.              (and (>= chk 65)(<= chk 90))
90.              (and (>= chk 97)(<= chk 122))
91.          )
92.          input nil
93.      )
94.   )
95.
96.   ;; TXINCR
97.   ;; Get Alphanumeric
98.   (defun txincr ( type index )
99.      (cond
100.        ((= type "Alpha")(chr index))
101.        ((= type "Numeric")(itoa index))
102.     )
103.  )
104.
105.  ;; Get Orientation
106.  ;; GODEF (GetOrient with DEFault value check subroutine)
107.  ;; Argument: a prompt string & a global variabe name.
108.  ;; Must be declare a global variable,gvar. Before use GODEF.
109.  (defun godef ( prmpt gvar / bye var)
110.     (setq bye "N")
111.     (while (/= bye "Y")
112.       (princ (strcat "\n  " prmpt ":<"))
113.       (princ gvar)
114.       (princ "> ")
115.       (setq var (getorient))
116.       (if (= var nil)
117.           (if (/= gvar "")
118.               (setq bye "Y")
```

รูปที่ 8 ไฟล์โปรแกรม TXTINC.LSP

```
                (princ "\n!!! Now not a default value. Must be input value, first.")
            )
            (setq bye "Y" gvar (angtos var 0 2) )
        )
    )
    gvar
)

;; Declare global varibles for used with GXXDEF function.
(setq #txt0 "" #txn "" #txh "" #txd "" #txalgn "" #prefix "" #suffix "" )

;;; Main Program
(defun c:txinc ( / olderr oldcmd oldblp justify cont flag txt0
                   txt align point inc pt0 txt
                 )
    (setq olderr *error* oldcmd (getvar "cmdecho") oldblp (getvar "blipmode"))
    (setvar "cmdecho" 0)
    (defun *error* ( msg )
        (if (not (member msg '("Function cancelled" "console break")))
            (princ (strcat "Error: " msg))
        )
        (command ".undo" "end")
        (setvar "blipmode" oldblp)
        (setvar "cmdecho" oldcmd)
        (setq *error* olderr)
        (princ)
    )
    (command ".undo" "begin")
    (initget 1 "Center Middle Right TL TC TR ML MC MR BL BC BR")
    (setq justify
          (getkword "\n  Center/Middle/Right/TL/TC/TR/ML/MC/MR/BL/BC/BR: ")
    )
    (if (= justify "Center") (setq justify "C"))
    (setq cont T)
    (while cont
        (setq #txt0 (strcase (gsxdef "First value" #txt0)))
        (if (txrng #txt0)
            (setq cont nil)
            (princ "\n Value must be any numeric or A-Z letters.\nTry again!")
        )
    )
    (setq flag (ascii #txt0))
    (cond
        ( (and (>= flag 48) (<= flag 57) )
            (setq txt (1- (atoi #txt0)) txt0 "Numeric")
        )
        (T (setq txt (1- (ascii (strcase #txt0))) txt0 "Alpha")
        )
    )
    (setq #prefix (strcase (gsdef "Prefix" #prefix))
          #suffix (strcase (gsdef "Suffix" #suffix))
    )
    (if (= #prefix "~")(setq #prefix ""))
    (if (= #suffix "~")(setq #suffix ""))
    (setq #txh (gddef "Height" #txh))
    (initget 1 "Uniform Random")
    (setq point (getkword "  Uniform/Random points: "))
    (cond
        ((= point "Uniform")
            (setq #txalgn (godef "Alignment" #txalgn ))
            (setq #txn (gidef "No. of Text" #txn))
            (setq #txd (gddef "Distance" #txd))
            (initget 1)
            (setq pt0 (getpoint "\n  Start point: "))
            (setvar "blipmode" 0)
            (setvar "lastpoint" pt0)
            (command "lastpoint" (strcat "@" (rtos (- #txd) 2 8) "<" #txalgn))
            (repeat #txn
                (command ".text" justify
                         (strcat "@" (rtos #txd 2 8) "<" #txalgn )
```

รูปที่ 8 ไฟล์โปรแกรม TXTINC.LSP

```
                                #txh 0.0
                                (strcat #prefix (txincr txt0 (setq txt (1+ txt)))
                                        #suffix
                                )
                            )
                        )
                    )
                    ((= point "Random")
                        (setvar "blipmode" 1)
                        (while (setq point (getpoint "\n  Draw point: "))
                            (command ".text" justify
                                     point
                                     #txh 0.0
                                     (strcat #prefix (txincr txt0 (setq txt (1+ txt)))
                                             #suffix
                                     )
                            )
                        )
                    )
                )
                (command "undo" "end")
                (setvar "blipmode" oldblp)
                (setvar "cmdecho" oldcmd)
                (setq olderr *error*)
                (princ)
            )
```

ตารางที่ 4 ตัวแปรระบบที่ใช้ในคำสั่ง TXINC

ตัวแปรระบบ	ความหมาย
BLIPMODE	ใช้ควบคุมการแสดงรอยกากบาท หรือ blip เมื่อมีการเลือกจุดหรือวัตถุบนหน้าจอ โดยค่าเป็น 0 และ 1 สำหรับการไม่แสดง และแสดง ตามลำดับ
CMDECHO	ใช้ควบคุมการแสดงพรอมต์ที่บรรทัด Command: ของคำสั่งใน AutoCAD เมื่อใช้ฟังก์ชัน command ในโปรแกรม AutoLISP โดยค่าเป็น 0 และ 1 สำหรับการไม่แสดง และแสดง ตามลำดับ
LASTPOINT	เก็บค่าจุดสุดท้ายที่มีการเลือกเกิดขึ้น การเรียกค่าจุดนี้ออกมาใช้งานให้ใช้เครื่องหมาย @ (at sign)

สรุป

สำหรับในบทความนี้ ได้นำเสนอคำสั่ง TXINC เพื่อลดการทำงานซ้ำ ๆ ที่ต้องค่อย ๆ เขียนชุดอักษรหรือตัวเลขเรียงแบบป้อนทีละค่า ให้เป็นงานของโปรแกรมที่เพิ่มค่าอักษรตัวถัดไป หรือเลขเรียงให้เอง แนวทางลดขั้นตอนการทำงานเช่นนี้ย่อมส่งผลให้ Productivity ที่ดี จัดเป็นการทำงานอย่างสร้างสรรค์ หรือที่เรียกกันทั่วไปว่า Smart draft